Australian Government
Department of Defence
Defence Science and
Technology Organisation

A Spoken Dialogue System for Command and Control

Adam Saulwick, Jason Littlefield and Michael Broughton

Command, Control, Communications and Intelligence Division

Defence Science and Technology Organisation

DSTO-TR-2754

ABSTRACT

A speaker-independent Spoken Dialogue Systems (SDS) is proposed as a more natural interface for human–computer interaction than the traditional point-and-click method. This report describes the objectives, development and initial implementation of an SDS within a prototype command environment. It includes design decisions and solutions to problems encountered during development as well as comments regarding ongoing and planned future research that have emerged through these activities.

RELEASE LIMITATION

Approved for public release

Published by

Command, Control, Communications and Intelligence Division
Defence Science and Technology Organisation
PO Box 1500
Edinburgh South Australia 5111 Australia

Telephone: (08) 7389 5555
Fax: (08) 7389 6567

AR 015-418
October 2012

APPROVED FOR PUBLIC RELEASE

A Spoken Dialogue System for Command and Control

Executive Summary

We report on activities undertaken in the research program 'Smart And Rapid Information Access for Joint Command and Control (C2)' one of whose goals was to enhance the spoken natural language (SNL) control of technologies and information for Deployable Joint Forces Headquarters (now Headquarters 1st Division) staff. The exploitation of SNL for C2 has multiple benefits, including improved 'naturalness' of interaction between headquarters staff and automated systems, such as those presented within a Livespace. Livespaces are dynamic, highly configurable, context-aware, smart-room environments designed for synchronous distributed collaboration [1]. For instance, a Spoken Dialogue System (SDS) can enhance human–computer interaction through the integration of voice-operated control of systems. However, this integration with 'traditional' interaction modalities needs to be done in a manner that is appropriate to the task and to ensure that it enhances, rather than hinders, a human's control of systems, access to information and workflow. Further, natural language-aware systems are arguably currently under-utilised in part due to the complexity of modelling and processing contextualised SNL.

This report describes our approach to, and actual development of, a speaker-independent SDS for control of devices and the presentation of automated briefs in the Livespace. We summarise here the engineering and development methodologies. These include requirements for implementing the SDS, such as SNL speech recognition and generation, as well as grammar and dialogue-management modules. We analyse the Natural Language Processing issues from an engineering and implementation perspective. This is because we believe that it is only on the basis of a functioning prototype that we can effectively assess its benefit to C2 and potential contribution to the capabilities of our clients.

The SDS described here permits user-initiated voice control, and state-querying, of devices such as computers, displays, lights and data-projectors in a technology-enhanced setting, such as the Livespace command environment currently developed within Command, Control, Communication & Intelligence Division (C3ID). Our SDS also enables staff to receive visual and audio briefs through synthesised speech at a time of their choice, and control them with their own SNL voice commands. The SDS integrates off-the-shelf Automatic Speech Recognition (ASR) and Text-To-Speech (TTS) output with sophisticated hand-authored natural language grammars which interpret SNL commands and queries. The human–computer interface is steered by a dialogue-management system that generates and coordinates real-time synthesised speech and device-update responses.

We have designed a modular, scalable, flexible, and robust SNL-aware system. To this end, we exploit both commercial and open-source tools and, where necessary, develop hand-authored components. Our SDS is designed to be increasingly sophisticated and responsive to users, and adaptable to relevant developments in language technologies and software developments, thereby providing clients with a state-of-the-art system that is powerful, efficient, reliable and commensurate with advances in command-centre technologies around the world.

Authors

Dr Adam Saulwick
Command, Control, Communications and Intelligence Division

Dr Adam Saulwick is a researcher in Human Interaction Capabilities Discipline. His current research focuses on theoretical and practical issues for modelling natural language for Human-Computer Interaction technologies and information fusion. He also conducts research in knowledge representation. Previous work in this domain focused on the formal representation of linguistic concepts in ontologies for data integration. His doctoral dissertation was a fieldwork-and-corpus-based grammatical description, text collection and dictionary of a highly endangered, polysynthetic Australian Aboriginal language.

Jason Littlefield
Command, Control, Communications and Intelligence Division

Jason Littlefield graduated with a B.Sc. in the School of Mathematics and Computer Science from the University of Adelaide in 2000 and joined DSTO in 2002. He commenced a Master of Science in Speech and Language Processing at Macquarie University in 2007. His research interests include automatic speech recognition, speech synthesis, and spoken dialogue systems that support collaborative activities.

Michael Broughton
Command, Control, Communications and Intelligence Division

Michael Broughton is a Human-Computer Interaction (HCI) specialist employed within the Human Interaction Capabilities Discipline of C3ID. He obtained a Bachelor of Computer and Information Science from University of South Australia and has broad experience within the field of HCI, including the application of speech and language technologies, experimentation, interaction within semi-immersive environments, and graphical user interface design. He was the task leader for JTW 04/195, leading a small team investigating the application of speech and language technologies within a C2 environment.

Contents

List of abbreviations/acronyms/initialisms

ADV	adverb
API	Application Programming Interface
Art	article
ASR	Automatic Speech Recogniser
C2	Command and Control
CDIFT	Coalition Distributed Information Fusion Test-bed
CFG	Context-Free Grammar
DJFHQ	Deployable Joint Forces Headquarters
DMS	Dialogue Management System
DSTO	Defence Science and Technology Organisation
FOCAL	Future Operations Centre Analysis Laboratory
GSL	Grammar Specification Language
HQ	Headquarters
HSI	Human-System Interaction
ICS	Intense Collaboration Space
Mod	modifier
MPP	Multimedia Presentation Planner
N	noun
NL	natural language
NP	noun phrase
Num	numeral
OBJ	object
OWL	Web Ontology Language
Part	particle
PC	Personal Computer
PNG	Portable Network Graphics
PP	prepositional phrase
Prep	preposition
Ptcpl	participle
Rel	relative
S	sentence
S'	'S-bar' a superordinate syntactic category, one level above S
SARINA	Smart And Rapid Information Access
SDS	Spoken Dialogue System
SID	Speaker Identification
SR	Speech Recogniser
SUBJ	subject
TCP/IP	Transmission Control Protocol and the Internet Protocol
THML	Talking Head Markup Language
TTS	text-to-speech
V	verb
VA	Virtual Adviser
VP	verb phrase
WCR	Word Correct Rate
XML	eXtensible Markup Language

1. Introduction

The DSTO Task JTW 04-195 entitled '*Smart And Rapid Information Access for Joint C2*' (SARINA) aimed to:

> *'… improve the interface to information for C2 by integrating language technologies into the human–computer interface for access, retrieval and presentation of information…* [and]*… investigate language processing and language generation approaches to allow more effective control by staff of the command support environment'*

A key activity of the task was to develop a user-initiated Spoken Dialogue System whose primary goal is to improve the naturalness of interaction between humans and machines. Our Spoken Dialogue System (SDS) interprets and responds to spoken natural-language commands and queries. The SDS translates spoken commands into system events. For instance, the command 'Turn on the lights!' triggers a light-activation event to turn on lights in an operations room. Alternatively, a spoken query, such as 'Are the front lights on?' triggers a synthesised speech response which informs the user of the current state of the lights; for instance, whether the particular lights are 'on', 'off' or 'dimmed'. Example synthesised speech responses to this query are 'Yes, the front lights are on' or 'The downlights are dimmed'.

With our SDS, the user can control and query a range of hardware devices in a room, including lights, computers, displays, and data-projectors, turning devices on or off and, in the case of certain lights, dimming. The SDS is also used to control and query software services, such as documents, network or web sites, and portals. This extends to opening, closing and moving a document or the desktop to another computer's display or screen, or even to projected displays. Commands and query examples are 'open a document on computer one', 'move the screen from computer one to the main display', or 'where is the screen from computer three?'. A separate grammar module enables voice-controlled presentation of a slideshow. Example commands are 'move forward three slides', 'skip this slide', skip the third slide', 'stop here', and 'go back to the beginning'. This module generates and controls a synthesised speech presentation of an existing slideshow's textual content. Example commands are 'start/stop (speaking the current) slide' or 'restart segment'. With this command the SDS synthesises the dot-point or paragraph text to speech.

The SDS enhances human–computer interaction beyond the traditional keyboard and mouse. We believe that the SDS enables commands and queries to be stated in a natural manner, and with increased speed and flexibility, than is currently obtainable with the keyboard and mouse. By 'natural' we mean a dialogue between human and machine that mimics the human-to-human interaction process. Given that the SDS responds to human commands and queries with appropriate synthesised speech (or other SDS system) responses, we believe that this gives users a sense that the Livespace is aware, and keeping track, of certain user requirements, such as room-state settings for presentation or collaboration modes. The SDS also records human–system dialogue interactions. These can be exploited by the user for retracing interactions. We believe that this further enhances the interaction potential not only between a user and an automated environment, but also

between humans. It does this by providing customised and relevant information on the Livespace environment status and real-time feedback to system requests.

Our SDS is user-initiated; it promotes natural interaction by allowing the user to command and query the system with voice, rather than follow system-led questioning to step through a pre-defined set of choices. In a system-led SDS, such as an automated taxi-booking application, human speakers must respond to and wait for spoken or textual prompts initiated by the system. This imposes a burdensome constraint on human–system interaction which we consider slow and overly cumbersome. Instead, we prefer a user-initiated system that is sophisticated enough to handle a human-led interaction which approaches the complexity of human-to-human dialogue. This approach is motivated by the aim to free users from common, yet arduous, aspects of system control, thereby allowing them to focus on core C2 tasks. Rather than distract or hinder users with time-expensive, technology-imposed, interaction obstacles, a goal of SARINA was to expedite C2 communications through stream-lined voice-control.

This report assumes some basic, general knowledge of computational linguistics and Natural Language Processing. See [2, 3] for an introduction to these fields.

2. The Spoken Dialogue System

The complexities of natural language (NL) typically necessitate development of an SDS in a restricted domain and with relatively limited language coverage. This reduces the search space and thus increases accuracy of utterance interpretation. This report focuses on our most recent user-initiated, English language SDS that has been developed for a room control system within a Livespace environment [1, 4].

The motivation was to address the requirement for voice-control of devices in Livespace environments. This provided the opportunity to implement a system in a constrained domain. The subsequent goal was to develop broad NL coverage for essentially a handful of commands. That is, we do not want the system to be limited to a precise, though NL, paraphrase of machine commands. Rather it should robustly handle and interpret multiple variations of NL formulations of the communicative intent. This was achieved with advanced NL and speech processing techniques.

This activity extends knowledge and experience acquired from our NL interfaces, such as those incorporated with Virtual Advisers in the Future Operations Centre Analysis Laboratory (FOCAL) [5, 6]. These earlier systems were based on template-style questioning and required verbatim query phrasing for system interpretation. In the current work, we developed grammars with broader coverage for the domain of Livespace room-control. The goal was to provide commands and queries to be expressed naturally and for the system to identify the user's intent, without the need to learn phrases by rote.

Within this room-control domain we explored numerous research topics. These include: (i) the application of linguistic techniques for the development of robust, flexible and optimised grammars; (ii) the application of dialogue management with conversationally acceptable responses; (iii) the provision for dynamic input of entities; (iv) the gathering of

corpora; (v) the integration of commercial-off-the-shelf speaker-independent automated speech recognition packages; (vi) an assessment of the efficacy or obtrusiveness of spoken NL commands/queries and system-generated synthesised speech feedback in a command environment, and finally; (vi) the provision of a suitable evaluation test-bed to measure the naturalness, coverage and robustness of the system.

2.1 Overview of the Spoken Dialogue System implementation

We implemented and demonstrated a prototype Spoken Dialogue System (SDS) in the Livespace environment of the Future Operations Centre Analysis Laboratory (FOCAL) [7]. This prototype was extended to other HQ C2 Livespaces such as the battlelab in Headquarters 1st Division. We developed 'commandable' language-aware electronic systems for the control of devices, and querying and presentation of information to enhance and stream-line processes in future headquarters environments.

We designed the SDS to be modular and comprised of the following general components shown in Figure 1 below: Speech Input, Natural Language Grammars, Dialogue Management System, Speech Output, and Domain Managers (a Room Manager and a Briefing Manager).

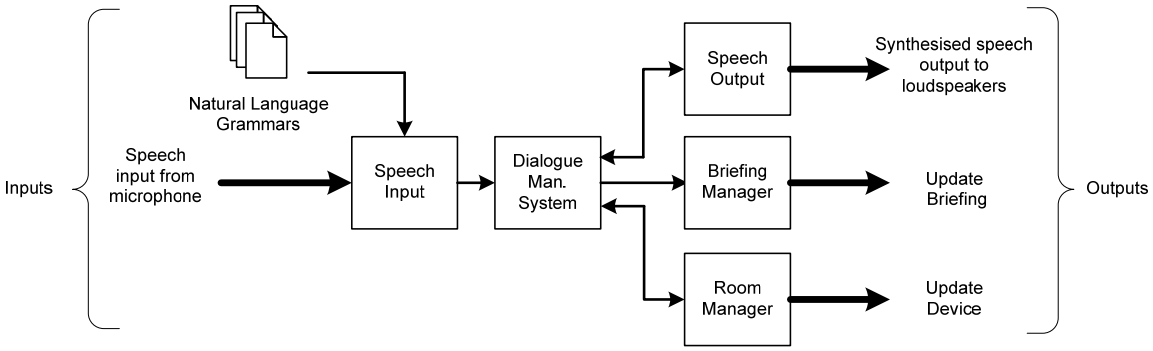

Figure 1: Spoken Dialogue System architecture showing components and data flow

We integrated a commercial Automatic Speech Recogniser (ASR) and a Text-To-Speech (TTS) System into the Speech Input and Output components. We developed the following three purpose-built, command grammars using Regulus[1] an Open Source toolkit for constructing spoken command grammars:

(1) Auto-brief grammar (version 01)
(2) a. Room Control grammar (version 01)[2]
    b. Room Control Specialised grammar (version 01)

[1] Developed since 2001 by a consortium, whose main partners have been NASA Ames, the University of Geneva, and Fluency Voice Technology, see also NASA on Regulus, and Rayner et al. [8].
[2] The grammars in (2) cover the same domain but represent significantly different implementation methodologies. The development of both grammars is used as a base-line for comparing system performance, see Sec 2.4.2.

These NL grammars are additions to the existing single FOCAL command-space grammar, and therefore represent a significant expansion in the natural language processing (NLP) capacity of Human Interaction Capabilities Discipline.

2.2 Design and Implementation Decisions

System requirements were key drivers in determining design and implementation decisions. The SDS needed to support three domains: Room Control, Automated Briefs and the Northern Quandary Scenario (see Section 2.8.6). In order for the SDS to be integrated with other systems developed at DSTO, it needed to be interoperable with, yet independent from, the agent-based and Livespace infrastructures. The agent-based infrastructure was developed for the Coalition Distributed Information Fusion Test-bed (CDIFT).[3] The Livespaces infrastructure provides a way for software services to communicate via an Enterprise Bus.[4] System flexibility, extensibility and robustness were important factors. We also sought to modularise grammar components to allow the inclusion of dynamic entities, such as customisable terms introduced at runtime.

The commercial Automatic Speech Recogniser (ASR) and Text-To-Speech (TTS) systems were chosen from Nuance Communications Inc. for a number of reasons: robustness; customisability via its Application Programming Interface; NL support; and availability through a research licence agreement. We chose the Regulus Grammar Development Tool-kit as grammar development software for five primary reasons:

(i) It compiles typed unification grammars into Context-Free Grammars (CFG).[5]

(ii) These CFGs are expressed in a format compatible with the Nuance[6] Grammar Specification Language (GSL), relatively easily incorporated into other components of the C2 Livespace system architecture, such as the Dialogue Manager module, the Text-To Speech module and the Real-World Entity Update module (see Section 2.5).

(iii) It is Open Source software and actively supported by a team of developers.[7]

---

[3] The CDIFT Interface is a package that implements communication between Agents via one of three methods CoABS (http://coabs.globalinfotek.com/), Elvin (http://www.mantara.com/about/) or XMPP (http://www.xmpp.org/).

[4] In computing, an enterprise service bus refers to a software architecture construct that provides foundational services for more complex architectures via an event-driven and standards-based messaging engine (the bus).

[5] 'A context-free grammar consists of a set of rules … each of which expresses the ways that symbols of the language can be grouped and ordered together, and a lexicon of words and symbols.' ([2] p. 327)

[6] Nuance is a natural-speech interface software for telecommunications, enterprise, and web-based systems. It is capable of speech recognition, speaker verification and identification and a degree of natural language understanding (see Nuance http://www.nuance.com/).

[7] Open Source software is computer software whose 'source code is available under a license (or … public domain) that permits users to use, change, and improve the software, and to redistribute it in modified or unmodified form. It is often developed in a public, collaborative manner.' (http://en.wikipedia.org/wiki/-Open_source_software). The opposite of Open Source software is Closed Source or proprietary software whose source code can only be modified by those with access to it, usually the creator. There are a number of general advantages of using open-source software. For instance, under usual conditions, Open Source software may be copied, modified and improved by anyone, thereby simplifying integration with other software applications.

(iv) It is used by other well-known organisations that require stable and dependable CFGs for demanding applications.[8]

(v) It is written in Prolog and is thus modular, extensible and able to be integrated with other system requirements.

2.3 Speech Input

2.3.1 Automatic Speech Recogniser Components

The Automatic Speech Recogniser (ASR) decodes the speech signal and transforms it into a textual representation. The ASR consists of a signal processor, a language model, multiple acoustic models, a dictionary, and a decoder. The signal processor extracts acoustic features from the speech signal. Acoustic features include formant[9] frequency, voice, and pitch. The language model calculates the probability of a word, given the sequence of words already detected.

Language models are typically grammar-based or probability-based. ASRs with grammar-based models usually require the rules of a grammar to be defined in terms of a context-free grammar represented in a predefined grammar format. The acoustic models are comprised of a digital representation of the acoustic features of the speech signal, including phone and word models.

Phone models are a set of acoustic features of recognised sounds for a particular language known as phones, such as the inventory of vowels and consonants. An example of an acoustic feature, recognised by a phone model, is voicing; that is, the difference between the voiced and voiceless sounds represented in written English with the symbols 'b' and 'p' respectively. Aside from the feature of [voicing], these phones have the same manner and place of articulation. Thus without the ability to differentiate features such as voicing an ASR would not accurately interpret the input signal.

Word models consist of a dictionary of words with pronunciations defined in terms of phones. Acoustic models are developed by performing phone alignment for hundreds of hours of speech-training data. The decoder compares the acoustic features, identified by the signal processor to those in the acoustic models, and produces phonetic representations with the most probable match. These are represented in a standard orthography (that is, writing system) for a given language; possibly accompanied by confidence scores, part-of-speech and prosodic features. Prosodic features include the melody, or intonation, of the speech.

We use an off-the-shelf ASR (from Nuance Communications Inc.) and integrate this into the system (see Figure 2).

[8] For instance, Regulus-built grammars are used by NASA's Clarissa, the first SDS to be used in space—on the international space station, and MedSLT, an Open Source medical speech translator developed at Geneva University.

[9] A formant is a resonant peak in an acoustic frequency spectrum. This is commonly used to differentiate speech sounds.

2.3.2 Automatic Speech Recogniser Performance

Optimal performance of the Automatic Speech Recogniser (ASR)[10] is crucial for the required performance levels of modules further in the system processing pipeline. For instance, low recognition by the ASR can result in ambiguities of syntactic-pattern allocation in the grammar. Thus the choice of an appropriate ASR is essential for each environment into which the SDS is installed.

There are over 80 task-related factors that affect the performance of an ASR, including: (i) dialectal variation; (ii) idiolectal variation; (iii) grammar complexity; (iv) parse algorithm; (v) hardware; and (vi) physical environment. We discuss each of these briefly below.

(i) Significant variation among the major dialects of English necessitates separate acoustic models for each of Australian and New Zealand, American, and British English.

(ii) Idiolectal variation (speech variation between speakers of a single dialect) resulting from differences in physiology, speech rate, gender, speaker's physical and psychological state, and experience all impact on ASR recognition rates.

(iii) The recognition task difficulty can vary depending on the size of the vocabulary and complexity of the grammar. ASRs with large vocabularies have a high recognition-task difficulty, and as a result, require further training by speakers to refine the acoustic models and improve recognition accuracy. These ASRs generate a speaker profile per speaker and hence are speaker-dependent. ASRs with small to medium vocabularies do not require additional training and are speaker-independent.

(iv) The type of parse algorithm has a bearing on the accuracy and performance of an ASR. Common algorithms are the hidden Markov model, Viterbi and the Baum-Welch. Most ASRs use a probability metric to select a highest likelihood of a match to the input signal.

(v) The amount of computer memory and processor speed significantly affects ASR performance. After extensive testing, directional headset microphones which are close-talking and noise-cancelling were found to provide the highest quality speech-signal, especially in noisy environments. However, headset microphones are sometimes obtrusive to wear and wired models tend to limit users' freedom of movement. Free-standing, directional, gooseneck microphones allow the user full freedom of movement. Unfortunately, gooseneck microphones tend to pick up speech from close neighbours, and so it is important to place them carefully to try to ensure that only one speaker is in range. In addition, these microphones are unsuitable in noisy environments. We use both types depending on environment.

(vi) Ambient and environmental noise, such as background conversation, as well as computer and projector fans, generally causes a significant reduction in the performance of the ASR. We removed noisy equipment in order to overcome this

[10] ASRs are generally categorised on different dimensions, including speaker-dependence, vocabulary size and speech continuity. Systems are classified as speaker-dependent, -adaptive or -independent. The vocabulary size refers to the number of recognisable words and speech continuity refers to whether utterances can be spoken in isolation or as continuous speech [9].

problem. This requirement has substantial implications on the overall cost and set-up requirements of a Livespace.

Our choice between the two basic ASR types was based on the difficulty of the recognition task. Typically, a speaker-independent recogniser is used for small to medium vocabulary, with connected and continuous speech recognisers, whereas a speaker-dependent recogniser is used for large vocabulary, with continuous speech recognisers. [11]

We chose a speaker-independent ASR that would handle input from both headset and gooseneck microphones. This was based on the following factors: our SDS has to operate with multiple users; our implementation environment is relatively stable (only medium-level background noise); and the domain is fairly constrained.

2.3.3 The Nuance components in our Spoken Dialogue System

The Nuance System (from Nuance Communications) forms an important component of our current SDS. It is comprised of several sub-systems: the speaker independent ASR (Nuance Speech Recognizer 8.5), a speaker identification module (Verifier 2.0) and a Text-To-Speech module (Vocalizer 4.0).

Nuance Speech Recognizer (SR) 8.5 supports 28 languages and dialects, including Australian and New Zealand English, and uses a proprietary Grammar Specification Language (GSL) to define context-free grammars. The Nuance SR includes tools for compiling GSL grammars and testing the compiled grammar package. Nuance also supports dynamic grammars that can be created and modified when the application is running. Nuance Vocalizer 4.0 has female and male voice packages for several languages and dialects including Australian English, British English, New Zealand English, Scottish English and American English. This is advantageous for tailoring to potential clients.

The Nuance System architecture uses a client/server model, where custom client software applications can be written, using a native API. The client connects to Nuance server applications using TCP/IP. The Nuance System consists of several software applications, monitored by a Windows Service called Nuance Watcher: a licence and resource manager, and four servers for speaker identification, TTS, compilation (which handles dynamic grammars), and speaker independent ASR. A useful feature of the Nuance System is that the Watcher automatically (re)starts with Windows thereby aiding the reliability of speech input and output. The section of Figure 2 inside the oval illustrates the architecture of the Nuance System integrated within our SDS.

[11] Commercial, speaker-independent ASRs include the Nuance Recognizer, CMU Sphinx, IBM WebSphere Voice Server, and Microsoft SpeechServer. Commercial, speaker-dependent ASRs include the Nuance Dragon NaturallySpeaking, IBM ViaVoice and Microsoft Speech Recognition Engine.

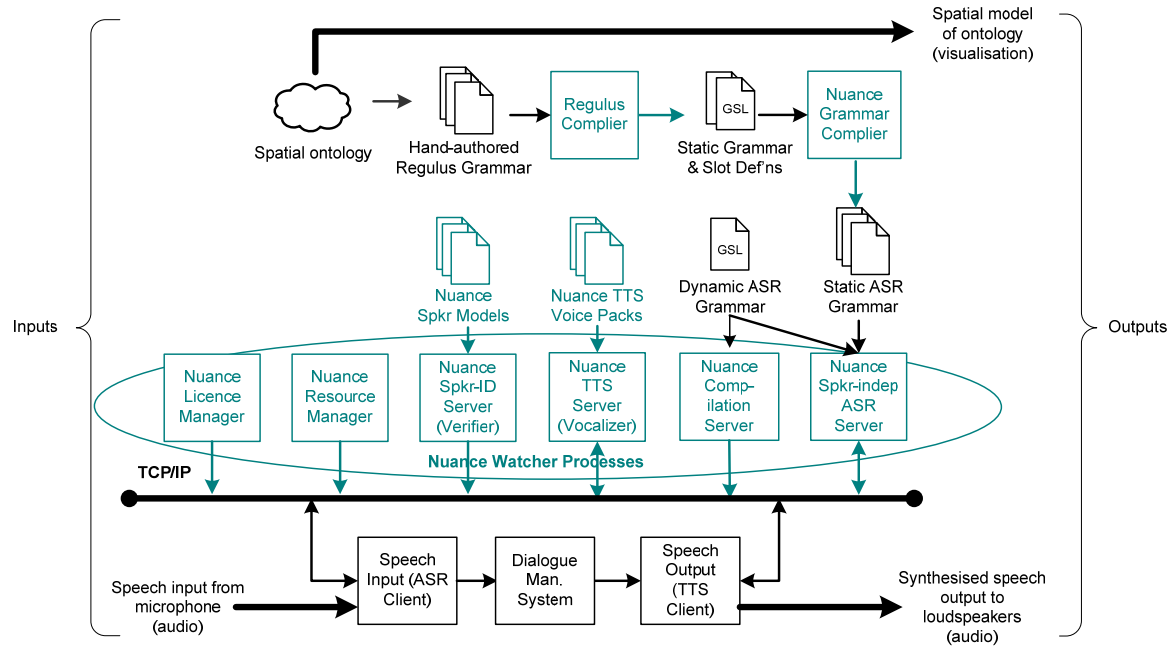

Figure 2: Nuance System integrated with Spoken Dialogue System components and Natural Language Grammar development process.

2.4 Natural Language Grammars

In general, a natural language grammar is required for rule-based interpretation of ASR output. The grammar needs to model a range of linguistic features found in NL and interface with the necessary components of the SDS. The complexity of the task at hand determines the sophistication of the grammar. For a system-led dialogue covering a very limited domain—such as a telephone-based interaction, targeting call preferences—a relatively small and simple grammar may be sufficient. However, given our domain (of user-initiated, command-and-query tasks for room control and automated-briefs) complex grammars were required. [12]

2.4.1 Linguistic requirements and challenges

Grammars capable of interpreting user-initiated dialogue contain a large number of basic linguistic features and structures. A full enumeration of these is far beyond the goal of this report. Nevertheless, an exemplification of one feature will demonstrate some of the general requirements and challenges. For the Room Control grammar, a basic requirement is differentiation of diverse utterance types, such as statements from queries (in linguistic terms, declaratives from interrogatives):

Declarative: simple present tense indicative declarative:
(3) a. 'Turn on the down-light in the ops room, please.'

[12] For readers unfamiliar with the concepts assumed in this section, see [2, 3] for an introduction to Natural Language Processing.

b. 'Move the screen forward from PC 1 to the left display.'

Interrogatives of various types:
Closed polarity, yes/no queries:

(4) a. 'Is the down-light on in the ops room?'
b. 'Is the screen from PC 1 on the left display?'

Open-ended wh-queries:

(5) a. 'Which down-light is on in the ops room?'
b. 'What (commandable component) is in the ops room?'
c. 'Where is the screen from PC 1?

To interpret these expressions, the grammar needs to recognise basic linguistic event types: one, two, or three participant events [10]. These can be usefully abstracted to *n*-place predication types, as in (6).

(6)

a.	one-place predicate:	be(x):	'the light is on'
b.	two-place predicate:	move(x,y):	'can you move screen forward'
c.	three-place predicate:	give(x,y,z):	'give me all information on X'
d.	pseudo three-place predicate:	move(x,y,z):	'move screen from PC 1 to left display'

The valency of the predication together with complements and possible adjuncts expresses the participant event-type.[13] *N*-participant events are captured in the grammars as event frames. Event frames are built into the grammar as independent constructional templates to which particular lexical items are associated through subcat(egorisation).[14] Examples of some subcat frames are given below. A one-participant *stative*-event frame can model a yes/no-query over the state of a device, such as a *light*, as in (7).

(7)

Be	Device	State
is	the light	(turned) on?

A two-place event frame models a transitive action, as in (8). Note that although (8) has two objects (Device and Location), one of them (the Location) is optional in terms of an effective Room Control command and so this command is modelled as a two-place event frame with an optional Location adjunct.[15]

(8)

Action	Device	Location
turn on	the light	(in the room)

[13] Valence refers to the number of arguments controlled by the predicate. For instance, the verb 'see' has two arguments a 'see-er' and a thing 'seen'.
[14] Subcategorisation refers to the specific syntactic requirements a linguistic predicate places on its arguments. For instance, the verb 'see' must have a subject noun phrase and a direct object noun phrase.
[15] An adjunct is an optional phrase, such as 'in the room'.

In contrast, a three-place event frame can model a ditransitive action (e.g., 'give', not in the current grammars) or a transitive action (e.g., 'move') with three obligatory objects (here Device, Source and Destination), as in (9a).

(9) a.

Action	Device	Source	Destination
move	screen	from PC 1	to left display

b.

Action	Device	Source
remove	display	from left display

Event frames and predication valency are examples of abstract foundational elements of the grammars. Without these, the syntactic patterns of spoken NL cannot be recognised, interpreted and integrated into the overall system.

2.4.2 Overview of grammar structure and development

In addition to other parts of the SDS, a structural feature of each incorporated grammar is its modularity; it is comprised of three core components: lexicon, syntax, and declarations. Each module is briefly discussed below.

2.4.2.1 The Lexicon

The lexicon consists of words and sometimes phrases structured in terms of word classes and sub-classes, i.e., noun, verb, adjective, adverb, numeral, pronoun, preposition, wh-word, etc. Each lexical entry is specified for its word-(sub-)class, and relevant semantic, grammatical, morphological and graphemic properties. For verbs, example properties are: overt form, action type, singular/plural form, valency, possible subject and/or object properties, and sub-categorisation frame. For nouns, example properties are: overt form, singular/plural form, semantic type, role type, modifier type. Macros are used to minimise redundancy of information in the lexicon. The lexicon must not only specify necessary function words but also cover the content words of the knowledge domain. In addition, the lexicon needs to handle semantic and functional information, such as synonymy (e.g., *to bank* vs. *to deposit*), polysemy (e.g., *bank* as a noun meaning 'depository' and 'stockpile'), homophony (e.g., *bank* meaning 'financial institution' and 'river edge or side') and idiosyncratic cases of word-class changing zero conversion (where a term is used as either a noun or a verb with no change of overt form, e.g., *to bank* ↔ *the bank*). Finally, it must contain appropriate output information for the dialogue manager (see Section 2.5).

2.4.2.2 The Syntax

The syntax stipulates the required and permissible syntactic structures for the specific C2 domain. These range in granularity from utterance level command and query structures to noun phrase constituency. A variety of phrasal and lexical constituent types are described, including S, V, VP, NP, PP, and Adjunct[16]. The syntax module needs to accommodate

[16] Abbreviations: S – sentence, V – verb, VP – verb phrase, NP – noun phrase, PP – prepositional phrase. An adjunct is a phrasal constituent that is not obligatory.

optional constituents and partially complete utterances. It encodes certain dependency relationships between constituents, such as frame semantics, phrase role and phrasal verbs. It also captures constituent order variation and associated idiosyncrasies. It needs to be aligned to the categories in the lexicon and their properties. As with the lexicon, it must contain appropriate output information for the Dialogue Manager.

2.4.2.3 The Declaration

The declaration consists of linguistic feature specifications for grammar elements, such as verb transitivity types, NP thematic role values, or utterance type values. These are associated with attribute-value lists for each feature. The named lists are associated with linguistic categories according to constraint requirements. Again, as with the other two grammar modules (Sections 2.4.2.1 and 2.4.2.2), the contents of the declaration must align with those of the lexicon and syntax.

Finally, an important component of the general formalism is—what is commonly known as—'feature unification'. In very basic terms, this means that each of the modules discussed in this section must contain attribute-value lists whose features unify with (that is, are compatible with) those features found elsewhere within the grammar.

The two NL grammars in (1) and (2a) are fully 'hand-authored'; they have been crafted from first principles drawing on specialist knowledge of general English grammar. In addition to hand-authoring, grammar (2b) is formed through the specialisation process, described in Section 2.4.4.

2.4.3 Hand-authoring natural language grammars

Development of NL grammars for a specific domain requires careful linguistic analysis of the relevant phonological, lexical and syntactic structures and their grammatical properties. These properties are then written into grammar modules, and iteratively tested and refined until recognition rates are sufficiently high.

Grammar-authoring is a cyclic process. An essential component in creating a hand-authored NL grammar for a user-initiated dialogue system is to include data-collection, assessment and revision phases. A sufficiently nuanced grammar should cover all possible utterances within the specified domain. For instance, in addition to the accurate representation of basic event types described in Section 2.4.1, there are a large number of possible expressions that speakers can use to express the same communicative intent. Examples of actual queries from a data-collection session (referenced by date, time and utterance number) are the (b) sentences in (10)–(11). Prior to data-collection, numerous variations of the anticipated (a) sentences were catered for in the grammar, but without the testing and revision phases, the (b) structures would not be parsed by the grammar and would thus fail recognition.

(10) a. Anticipated: 'Which screen is on the right display?'
b. Actual: 'What is being displayed on the right display?' (2007-05-31-1540-utt17)

(11) a. Anticipated: 'Where is the screen from Dream PC one?'
b. Actual: 'Where is Dream PC one forwarded to?' (2007-05-31-1540-utt28)

A detailed analysis of the linguistic differences between the above anticipated and actual utterances is not pursued here. However, the difference between the two (anticipated versus actual) utterances highlights the important role of the iterative development process. Nonetheless, this process involves a trade-off between producing a sufficiently nuanced grammar (with the greatest coverage of naturally occurring utterances) on the one hand, and forestalling the creation of a grammar which is sensitive to a myriad of possible yet unwanted expressions on the other hand. An over-sensitive grammar will interpret too many possible ambiguous structures, resulting in lower, and slower, recognition rates. This area requires further work on optimization.

We now summarise the properties of the two grammars.

2.4.3.1 Auto-brief grammar

Auto-brief (version 01) is a command grammar for a generic slideshow brief. An overview of the features of the Auto-brief grammar is given below:

Features and coverage:
- Imperative commands for basic action verbs
- Common nouns, differentiating number
- Adjectives, differentiating two premodifying tiers (ordinal versus colour qualifying) and postmodifying (cardinals)
- Articles, including definite and indefinite
- Demonstratives, including locational deictics
- Adverbs, differentiating direction and some basic locational deictics and including intensification
- Prepositions, including source and goal

Special features:
- Permits limited anaphora
- Handles full ellipsis of VPs and NPs
- Provides keys for linkage to action commands
- Compiles to Nuance GSL grammar

Problems to be addressed:
- Slight over-generation, probably resolved by adding further selectional restrictions

Some examples of coverage are:

(12) a. '(Can/could you) start/restart/play/pause/continue/stop/end (the) slideshow (please/thanks/thank you very much)'
b. '(Move) forward/backward *n* slide(s)'
c. 'Next (slide)'
d. 'Skip the next *n* slide(s)'
e. 'Skip to the very first/last slide'
f. 'Skip to the end'
g. 'Jump/go/continue/forward to slide *n*'
h. 'Repeat slide/segment (*n*)'
i. 'Restart from/at slide *n*'

Hand-authoring this grammar meant that full control over permissible grammatical structures was possible with great flexibility. This did not lead to an appreciably excessive recognition problem.

2.4.3.2 Room Control grammar

Room Control (version 01) is a command and query grammar for control of devices in a Livespace or similar space. An overview of the features of the Room Control grammar is given below:

Features and coverage:

- Imperative commands for basic room control verbs, including special handling of particle verbs 'switch/turn X on/off'
- Common nouns, differentiating semantic values based on light location and display location; switchable devices as well as moveable and openable/closable entities. Also distinguishes number: singular and plural
- Adjectives, differentiating relative locations, such as 'front, back, rear, centre', etc. and numerals (ordinals and cardinals)
- Prepositions, including location, source and destination
- Politeness expressions, pre- and post-modifying, such as 'could/can you, please' and 'thank you (very much)'

Special features:

- Handles particle verbs, permitting alternative syntactic structures:
  'turn on X' and 'turn X on'
- Accommodates multiple syntactic frames:
  (ADJUNCT) SUBJ V OBJ1 OBJ2 (ADJUNCT)
- Optional modifiers and adjuncts
- Handles synonyms and short forms: demo(nstration) room, (op(eration)s) lab
- Optional politeness expressions
- Comprises placeholders for dynamic NP expressions of types: Device, Location, Source, Destination
- Compiles to Nuance GSL grammar

The constituent structure of the Room Control grammar is given in Figure 3. This represents a maximal syntax for declaratives and interrogatives. Note that dependency relations in the constituency have been built into the grammar, such that not all lexical level entries can co-occur.

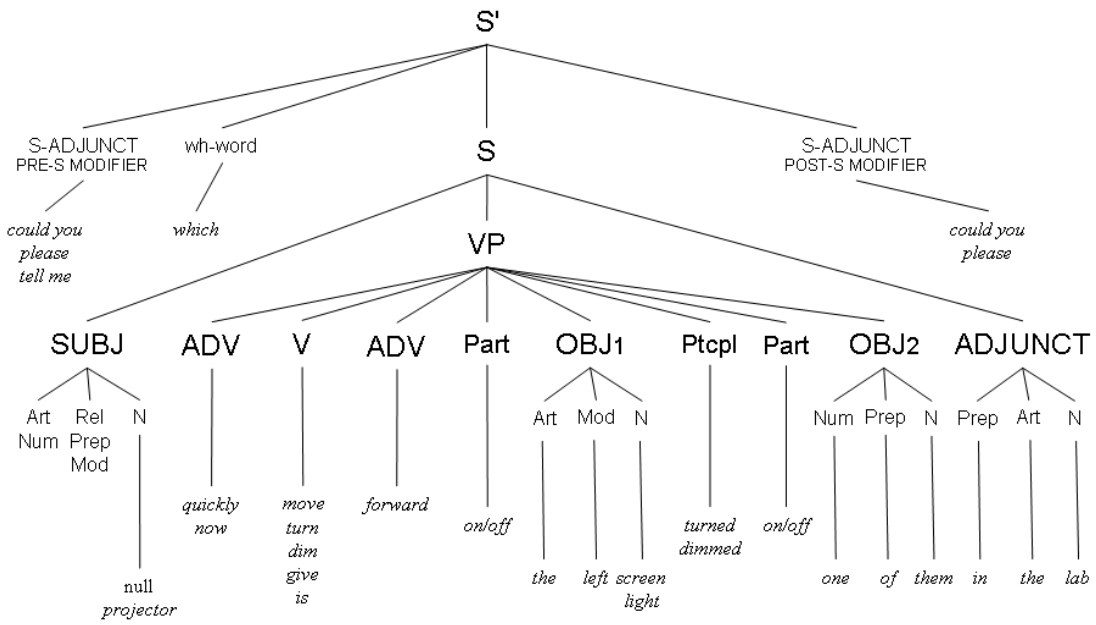

Figure 3: Constituent structure for Room Control grammar

Abbreviations of category labels used in Figure 3 are: ADV – adverb; Art – article; Mod – modifier; N – noun; Num – numeral ; OBJ – object; Part – particle; Prep – preposition; Ptcpl – participle; Rel – relative; S – sentence; S' – 'S-bar' a superordinate category, one level above S; SUBJ – subject; V – verb

2.4.3.3 Data collection

We carried out three data-collection sessions to record NL utterances which could be used to test and revise grammars. Rather than dictate the precise statements to be uttered verbatim, participants (other Task members) were given a one-page handout with information on the types of objects that could be referenced, and the types of activities or states that could be uttered with regard to those objects. This handout (shown in Figure 4) also gave some example sentences.

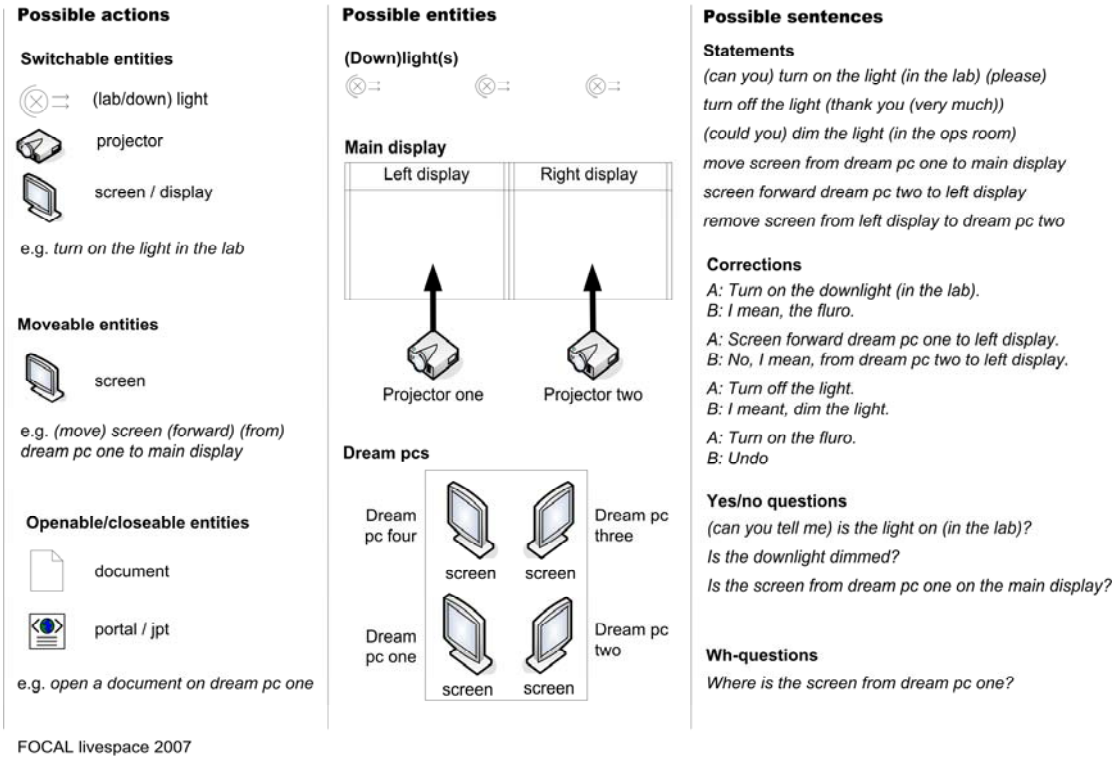

Figure 4: Handout given to participants prior to data collection

The aim of this handout was to give participants a general overview of the types of possible utterances and to assess two things. The degree to which:

(i) a handout of this kind was adequate in making explicit the linguistic possibilities of the Room Control grammar;

(ii) the participants' expectations of what they could say—based on information in the handout—were covered by the grammar.

Initial observations suggest that the handout was adequate for the purpose of giving participants a general understanding of a voice-operated room command system, but deficient in terms of providing enough feedback to participants in a spoken data-collection activity which was not implemented in an actual command-room. For instance, although a spoken command would parse in the grammar, and provide a system response, participants had no feed-back as a point-of-reference for determining the state of an entity. For future data-collection sessions it would be useful to have moveable tokens so that participants could keep track of device locations. Alternatively, data could be collected in the actual implementation environment (i.e., a Livespace); however, this would add an additional level of complexity to the activity as it would require the SDS to be fully integrated with the system-update modules. An alternative approach would be a Wizard of Oz mechanism and this may be attempted in future.

In addition to highlighting the need for revision to broaden grammar coverage (Section 2.4.3), the collected data will also be used for systematic regression testing. We do not

discuss regression testing here; suffice it to say that it is used to increase recognition performance.

2.4.3.4 Dynamic grammars

A further goal of our implementation is to permit the addition of user-supplied entity names to the grammars at runtime (i.e., during operation, commanders could add their preferred personal- and/or device-names to the SDS for name customisation). This feature, called 'dynamic grammars', requires a degree of structural flexibility in the core hand-authored grammar to accommodate unsupervised additions to particular sub-parts of the lexicon. Though this could potentially cause problems for other parts of the system, we feel that it will provide an added sense of natural interaction with the SDS and become a useful system enhancement.

A challenge for dynamic grammar capability was to make the core grammar robust enough to handle the addition of new words to a component of the lexicon, while maintaining full functionality. We achieved and demonstrated the successful but rudimentary implementation of the dynamic component in both the hand-authored and specialised Room Control grammars (Section 2.4.4). The dynamic subcomponent is instantiated in the required places in the compiled GSL grammar and provides a means of mapping grammar-independent entities into the syntax of the grammar in real-time. To date, this has not been implemented to the point where it could be used in the data-collection sessions, but is an area targeted for future research and development.

2.4.4 Grammar specialisation

Grammar specialisation is the process of generating a GSL grammar from the combination of a hand-authored, domain-specific, lexicon with a general pre-existing (English) Grammar module. A claimed strength of specialisation is that it is possible to produce a grammar recogniser through derivation from a single general feature grammar ([8] pp. 149-173). In addition to the individual lexicon, this process requires the establishment of an appropriate corpus. There are, however, some problems with the specialisation process which need addressing:

- The efficacy of specialisation is contingent on creating an adequate formulation of special rules (known as Explanation Based Learning rules) which stipulate the relevant syntax for the specialised grammar with respect to the corpus. This is not discussed further here.
- It requires a training corpus of significant detail. This means that any adjustment of the grammar needs to be done in parallel to a modification of the corpus.
- The overall flexibility of the specialized syntax is to some degree constrained by the general English grammar from which derived grammars are specialized.
- Modification to the general Grammar Module from which the specialized grammar is formed is complex and apt to compilation failure.

A specialised version of the Room Control grammar was created, and compiled into GSL form with the aid of a domain specific lexicon and a substantial hand-built training corpus. Its coverage includes a list of simple command verbs for switching basic elements of an

operations room on or off, such as lights, local PCs and data-projectors. It can also parse commands for moving digital elements of an operations room such as the image on a local PC (a document) and a large projected data-display. We do not address further here the complex issues raised concerning specialised grammars.

2.5 Dialogue-Management System

The Dialogue-Management System (DMS) is the component of the SDS which manages and responds to the language output from the grammar. It provides the following services: (i) feedback to users as synthesised Text-To-Speech (TTS) or in text; (ii) information for triggering system events (e.g., turning on lights or allocating computer output signals to particular displays); (iii) ongoing documentation of the dialogue flow, and; (iv) an entity state-update.

A key aspect of the DMS is its responsiveness to users. By generating real-time TTS feedback, users receive NL responses which coincide with the visual actualisation of their commands, e.g., Command: 'screen forward to the front display' – TTS response: 'OK'. If a TTS response is considered too invasive, users can also select a response chime. In the case of queries, users receive real-time TTS responses reflecting the current system-state, e.g., Query: 'Where is the screen from desk one?' – TTS response: 'Desk one is on projector three'.

There are three modules to the DMS: an input manager, a dialogue manager and an output manager, see Figure 5. The DMS is integrated with a Regulus Server from the Regulus Grammar Development Toolkit via TCP/IP sockets. The Regulus Server provides speech input and speech output functions by accessing ASR and TTS servers via TCP/IP sockets, see also Figure 5.

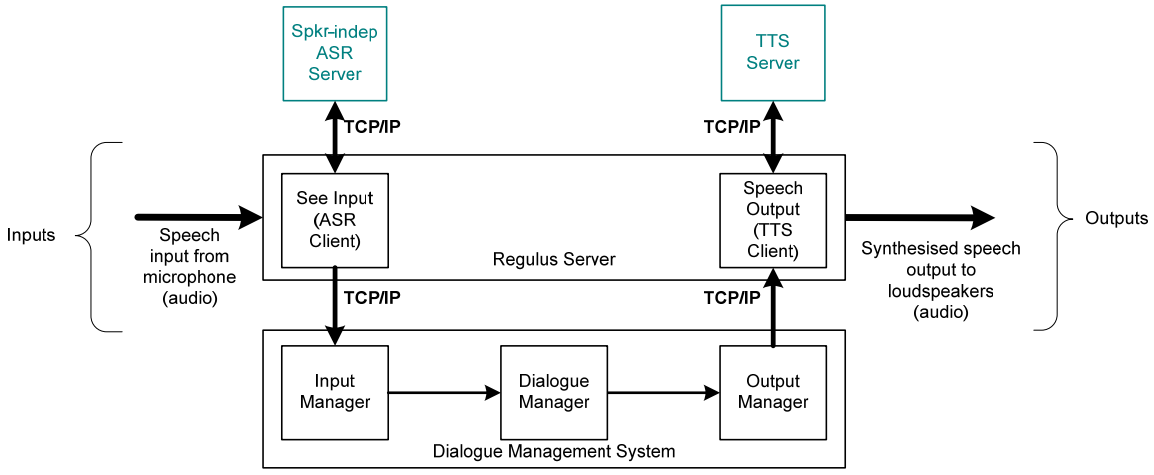

Figure 5: DM integration with Regulus Server providing Speech Input and Speech Output functions

The dialogue manager permits certain additional natural human interactions. It recognises and handles common conversational hiccups, such as false-starts, corrections, missing elements and even 'undo' commands. For instance, by prefacing a command with 'No, I

meant…' an operator can correct a system event by uttering a new, overriding command in a following statement, e.g., Command 1: 'Forward the screen from desk one to the front!' Command 2: 'Ah, no, I meant, the screen from desk two to the front!'. Even after an event has been actioned, the dialogue manager will replace the first command with the second so that only the data from desk two is displayed on the front. Note that in Command 2 the word 'forward' was missing. We have designed the dialogue management system to accommodate this type of ellipsis. Finally, it is also possible to retract previous commands by moving through all those previously uttered; the utterance 'undo (that)' results in the reinstatement of the immediate, prior state. With this 'undo'-command the operator can return stepwise to any prior state.

2.6 Domain Managers

2.6.1 Briefing Manager

An automated briefing is a slide presentation generated from slides and notes, and narrated by a digital representation of a human talking head (a virtual adviser) integrated with the TTS system. The briefing manager interprets briefing commands from the SDS and orchestrates actions for the digital-video presentation software (such as Media Player), the virtual adviser and TTS components, such as starting and stopping the presentation, and navigating through the presentation slides in response to spoken user commands.

Briefing slides with narrative notes in the form of a Microsoft PowerPoint presentation first need to be manually created. Prior to presentation by a virtual adviser, the slides are then automatically converted (with a Visual Basic Macro) into automated briefing data. The automated briefing data consists of a file that includes an ordered list of clips (previously a PowerPoint slide) and a folder with a Portable Network Graphics (PNG) image file for each slide. Its format is either Extensible Markup Language (XML) or Talking Head Markup Language (THML). Each clip includes metadata such as clip ID, comment, and segments with content (slide PNG path, narrative notes, etc.).

Spoken NL commands are recognised by the speech input component as speech-acts with constituent structures based on a parser which utilises the hand-authored auto-brief grammar described in Section 2.4.3.1. Each constituent is allocated a value, called the 'logical form'. For instance, the utterance 'move back one slide' has the logical form: '[command [verb [action, move] [adv [direction, back]]] [np [[cardmod, 1] [device, sshSlide]]]]', e.g., the value for 'action' is bound to 'move'. In addition to associating linguistic categories, such as verbs and nouns with instances (e.g., [action, move]), the logical form also stipulates syntactic relations of the sentence or phrase structure (e.g., that the numeral 'one' and slide-show slide form a noun phrase constituent). The dialogue management system interprets the logical form as a speech-act and generates briefing commands such as '(move) back one (slide)'. The briefing manager coordinates actions for corresponding briefing commands with the Media Player, the virtual adviser and the TTS system. Figure 6 illustrates the automated briefing-system components and workflow.

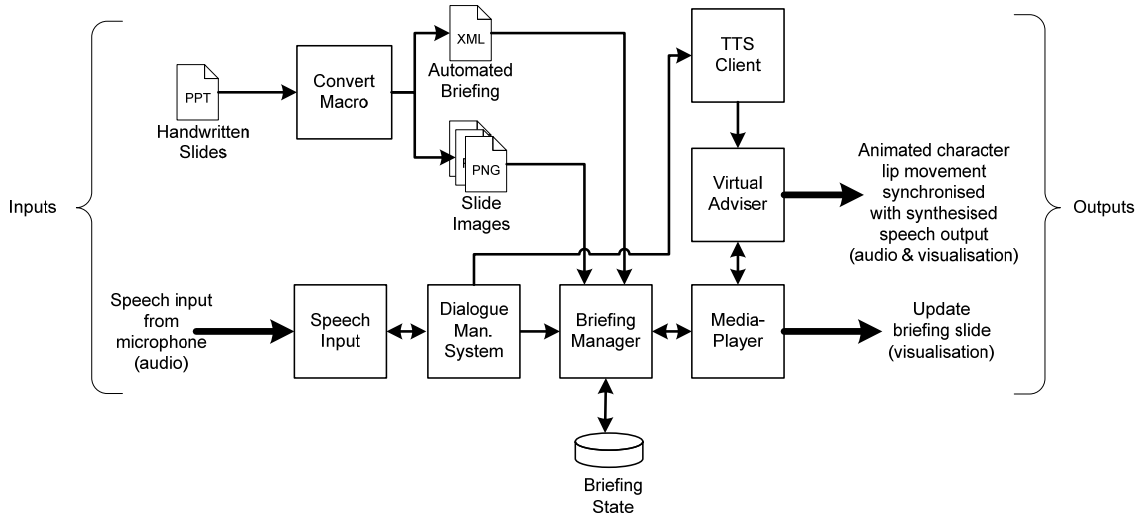

Figure 6: Automated Briefings system overview including Spoken Dialogue System and Automated Briefing Generation components

2.6.2 Room Manager

The room manager provides an interface between the agent-based SDS and the bus-based Livespaces. The scope of the Livespace initially included software services controlling devices such as lights, data-projectors, displays, computers, and audio and display switches in a room. Each Livespace service publishes property-value pairs and provides a way to control a device by setting (and getting) its property values.

The Room Control grammar, DMS, and Room Manager coordinate to support spoken NL commands and queries. Using the Room Control grammar described in Section 2.4.3.2, the Speech Input component recognises a speech-act's constituent parts, e.g., 'Turn the downlights on in the lab!' is transformed into the logical form with the following attribute-value pairs: '[[s_type, imp], [utterance_type, command], [action, switch], [spec, def], [device, downlight], [device_type, light], [onoff, on], [prep, in], [spec, def], [location, room]]'. The DMS then resolves abstract nouns to the name of the actual device which the Livespace services control.

In the case of a spoken NL command, the DMS maps the speech-act into a list of Livespace service property-value updates, e.g., [:request-type update :entity-name 'lab.downlights' :property 'level' :property-value '100']. The list of Livespace service property-value updates is handled by the room manager, which sets the values of the corresponding Livespace service properties.

In the case of a spoken NL query, e.g., 'Is the lab downlight on?', the DMS accesses a local-store of supported Livespace service states, generates a NL response, e.g., 'yes', and passes this to the TTS system.

We developed a messaging protocol for Livespace service property-queries and for updates between the room manager and other components of the SDS. The interface between Livespace and the SDS is presented in Figure 7.

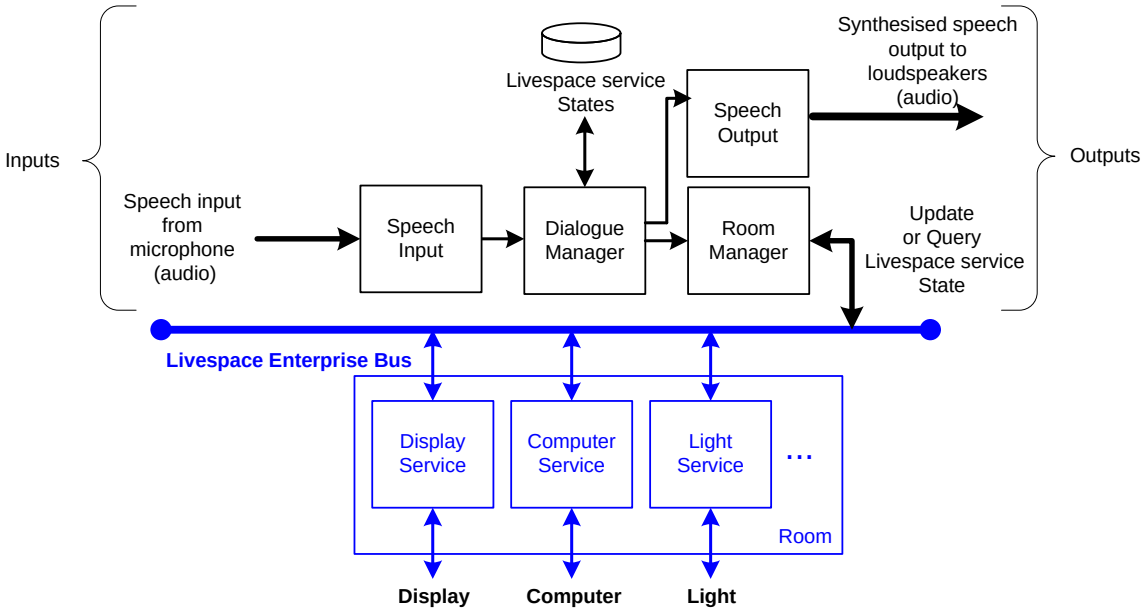

Figure 7 System components connecting room manager to Spoken Dialogue System and Livespaces

2.7 Demonstrations

An early version of the SDS which used the Auto-brief grammar, and associated Dialogue Manager, was demonstrated during Exercise Pozieres Development in October 2006. Current versions of the Room Control grammar are demonstrated as part of a capability demonstrator of divisional technologies called the C3I Integrator.

2.8 Ongoing and Future Work

Some aspects of work described thus far are relevant to future research and development, and are discussed in the following subsections.

2.8.1 Dynamic-grammar Integration

As discussed in 2.4.3.4, the ability to add—at runtime—new entity names for existing devices was achieved to a rudimentary level. We plan to develop the dynamic functionality to permit the incorporation of other word types and, possibly, elements of dynamic syntax, thereby extending the degree of coverage to user-determined structures. This would require analysing the range of possible dynamic types and proposing an appropriate user interface for inputting information with certain syntactic structures.

We demonstrated dynamic-grammar integration for entities of the type 'light' with the Room Control grammar. This involved four components of the SDS. Dynamic-grammar place-holders were added to the static-grammars. A dynamic-grammar in GSL format was automatically generated at runtime which used dynamic names for lights. We added the Nuance compilation server application, which compiles dynamic grammars at runtime, to the list of applications monitored by the Nuance Watcher. We also developed a purpose-

built speaker-independent ASR client application that incorporates both the static and dynamic grammars. Dynamic-grammar integration into this component still needs to be implemented for the current SDS as well as support for other types of entities.

2.8.2 Grammar Performance Optimisation

A second area of further research concerns an assessment of the overall performance of the SDS. To date, performance tests of only subparts of the SDS were undertaken. We intend to carry out systematic regression testing to assess the robustness, viability and extensibility of hand-authored versus specialized grammars. Data collected from command and query sessions will be used as a baseline for tuning grammars with the aim to increase the performance of both ASR and Dialogue Management components.

2.8.3 Dual-Type Automatic Speech Recogniser

A Dual-Type ASR is a multi-pass ASR system that incorporates both speaker-independent and -dependent ASRs in an effort to improve the robustness of an SDS (see [11]). The components of the system's design included an independent utterance recorder, speaker-independent and –dependent ASRs, speaker identification, an error detector and a recognition result reconciler.

Many of these components still require implementation before planned experiments can be performed to determine which system design provides the best performance and user satisfaction. For instance, any time delay between the completion of a spoken utterance and the action-execution detracts from user satisfaction and confidence in the SDS. The addition of an audio matrix switch to the audio system will reduce the time delay to some degree by allowing simultaneous speech input from microphones to the utterance recorder, speaker-independent and –dependent ASRs, and speaker identification systems.

2.8.4 Improving ASR Recognition Results by Majority Voting

As part of the Dual-Type ASR described in Section 2.8.3, a system for combining, or choosing between, recognition results from two ASRs is another avenue for potential optimisation. This needs to be assessed to ascertain whether it is at least as accurate as the results from a single ASR. We are aware of a similar system for Machine Translation called DEMOCRAT, developed at Macquarie University [12]. In collaboration with van Zaanen, we investigated majority voting techniques to combine recognition results [13].

The initial plan was to use three commercial ASRs to transcribe a speech corpus of 1680 wave files, and compare the n-best list for each with a reference corpus. Unfortunately, two of the three ASRs provided results with word-correct rates (WCR) too low for use by a majority voting system. We propose to use the n-best list of recognition results from a single ASR.

An ASR transcribes spoken utterances into a ranked n-best list of recognition results, and by default, provides the recognition result ranked highest, called alt_0. However, based on informal observations, when the ASR produces less than 100% accuracy, the n-best list often contains a better alternative. An experiment to quantify this found that 37% of the time the n-best list provided a result better than the default alt_0.

On the premise that post-processing the *n*-best list could provide a more accurate result, a system was developed that tries to improve ASR recognition results by majority voting [13]. A second experiment to determine if post-processing with this majority voting system improved recognition result accuracy found that the mean WCR improved only marginally (*WCR = 76.3%*) over the highest ranked recognition result (*WCR = 75.9%*) [13]. Statistical analysis showed that the difference was statistically significant. Future work will pursue the multiple ASR approach to performance improvement.

2.8.5 Speaker Identification System

Speaker identification (SID) systems automatically determine the speaker of an utterance from a pool of registered people. SID can have a variety of advantages, such as rapid authorisation at system start-up, user-customisation, and the possibility of improved ASR rates. Higher ASR rates may be possible if a speaker-dependent grammar can be used a once a speaker has been identified.

SID is achieved by extracting acoustic features from the audio signal and comparing them with reference models for registered speakers; a correspondence results in the identification of a speaker.

Queensland University of Technology (QUT-UBM) developed a SID agent that uses a Universal Background Model SID system. Preliminary testing of this gave positive results, with different utterance lengths of 0.5, 2.0, 4.0 and 8.0 seconds producing error rates of 57%, 18%, 10% and 6%, respectively. However, over the course of 2005–2007, the system's performance deteriorated significantly. To date, an investigation of the cause of degradation revealed several problems in the way the system was implemented, some of which were rectified.

First, the SID input audio was recorded with a Nuance ASR and was poor quality and sometimes incomplete. This was caused by narrow network bandwidth, and rectified through reconfiguration; audio files were saved on local rather than remote drives. Second, the audio format between the Nuance ASR and the SID may be incompatible. Due to changes in lab equipment, there is a probable mismatch between training and test audio. Third, environment noise and reverberation characteristics have changed with the installation of new data-projectors and a new Livespace framework. In sum, SID audio training data needs to be adjusted in consideration of the particular physical environment along with new speaker models.

An alternative solution would be to integrate the SID system with the commercial speaker Verifier available from Nuance. This requires further investigation.

2.8.6 Natural Language Queries for a specific scenario

The previous agent-based SDS supported a limited set of NL queries whose content was based on a fictional scenario. A Regulus specialised grammar was developed for use with the Nuance ASR. The SDS recognises speech-acts and generates communicative goals, which are passed to the Multimedia Presentation Planner (MPP). The MPP produces a discourse strategy, a media selection strategy, and a presentation plan. The presentation plan is handled by a local rendering agent that orchestrates the display and timing of

media types to the virtual adviser (VA) and Media Player components. The VA is embodied by an animated character with lip-sync for synthesised speech from Text-To-Speech. The Media Player displays text, still images, and video. We plan to integrate the scenario grammar, MPP, VA and Media Player into the current SDS.

2.8.7 Integrating a structured knowledge base

We demonstrated, to a rudimentary level, the ability to exploit a structured knowledge representation of real-world entities (in the form of an OWL ontology of the Livespace environment) for off-line development of the lexicon. This is valuable for systematic development of lexica, and could be exploited for visualising parts of the environment to which the grammar refers (see Section 2.8.8). For instance, based on the entities in an ontology, operators of the Livespace could view on their displays an image of the room, showing entities available for control via voice. Further research is required in this area, including an appropriate input technology, visualisation, and streamlined integration.

2.8.8 A comprehensive Dialogue System

Finally, we see the SDS as a part of a comprehensive and modular Dialogue System. This is motivated by our adherence to the principles of extensibility and adaptability, which we hold as necessary for implementation in environments with a diverse array software and hardware. Modularity should also provide some general robustness against potential system break-down that may result through version incompatibilities, loss of technical support for commercial programs or associated compatibility issues. While the current SDS has achieved full through-put in terms of processing spoken language, providing feedback via TTS output and actioning state updates, we still consider it only partially complete. A fuller, or possibly 'complete', system (depicted in Figure 8, with extensions to the currently implemented modules shaded in 10% grey) would be able to interpret a variety of natural multi-modal inputs and generate various multi-modal outputs.

The motivation for interpreting and delivering additional modalities maximises information transfer for human operators by exploiting other communication channels that are un(der)utilised in a Dialogue System that is constrained to the speech channel. On the input side, we envisage the combination of keyboard, gesture, gaze, and (multi)point together with spoken input. Similarly, on the output side, we envisage the combination of synthesised speech with various visualisation possibilities, such as textual and graphic representations of a physical environment, animated characters and visual briefs. The merit of avatars and other visualisations is an important area of ongoing research in the broader systems integration community.

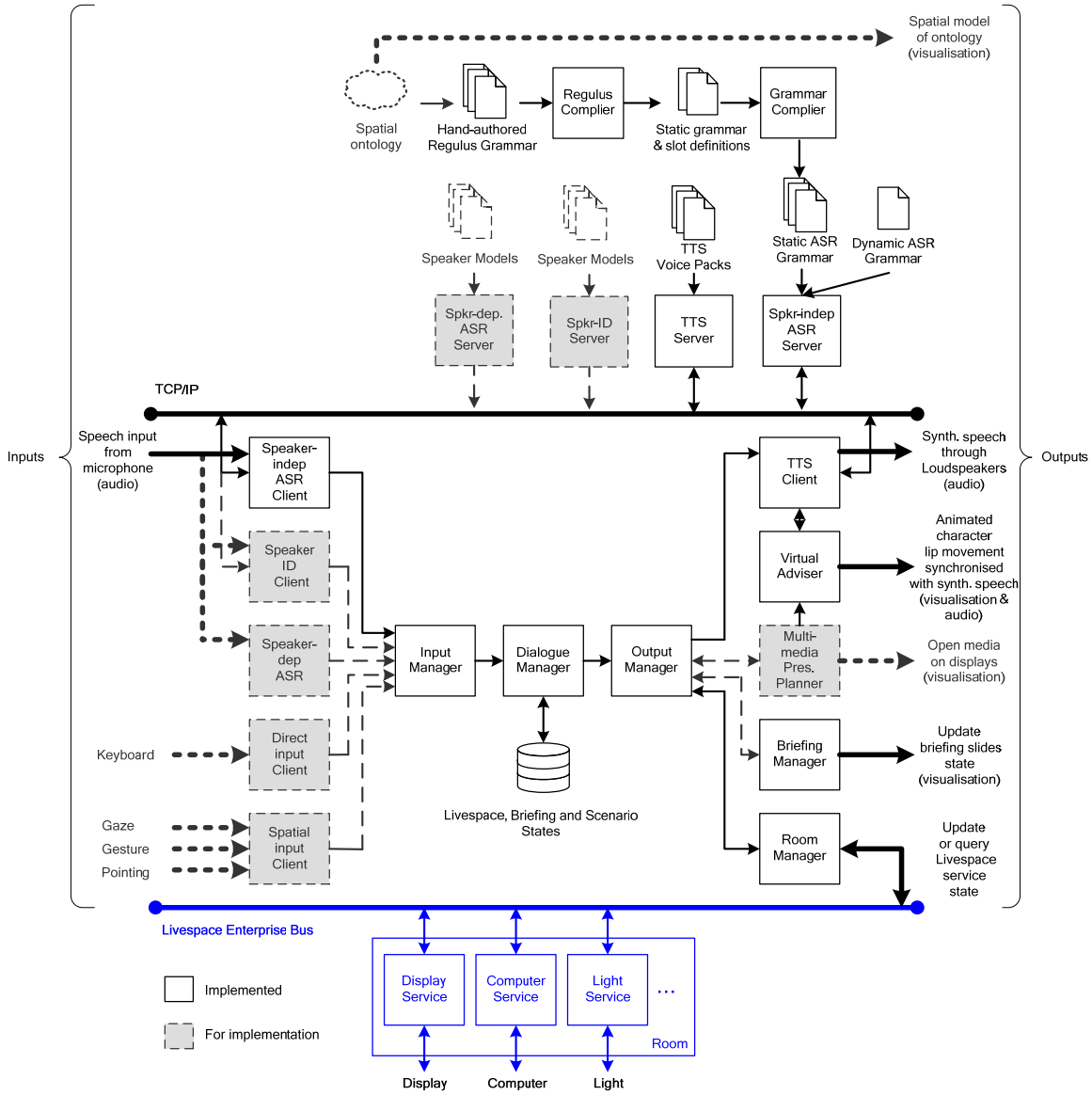

Figure 8 Design for a comprehensive Dialogue System

3. Conclusion

We developed and implemented in the Livespace environment a speaker-independent Spoken Dialogue System (SDS) that interprets user-initiated communicative intents in specific domains and provides operators with voice-control over display, media and room components.

The SDS is highly modular. We integrated various off-the-shelf (commercial and open-source) packages with hand-authored modules to construct a complex system which was tested and demonstrated in the Livespace ICS zone of the FOCAL and Intense Collaboration Space laboratories at DSTO-Edinburgh. The SDS is designed to be increasingly sophisticated and feature-driven in response to client requests and research

foci. Specific areas of research and development are the expansion of the core grammars and dialogue management modules to increase coverage and to handle discourse-level features of natural language, such as anaphora resolution[17] and the stream-lined integration of dynamic-grammars. Furthermore, expansion of the coverage of each grammar, as well as the integration of other types of knowledge through further exploitation of ontologies, together with optimised ASR and grammar modules, will increase the system's overall responsiveness, flexibility and robustness. This will lead to a mature and state-of-the-art SDS that is powerful, efficient, reliable and commensurate with advances in command-centre technologies around the world.

[17] Reference to an entity that has been previously introduced in the discourse with a substitute term is called anaphora. Anaphora resolution is the identification of the correct correspondence between the previously introduced entity and the substitute. For example, in the sentence *Mary saw John at the office, while he was reading*, the previously introduced term is the noun *John* and the substitute is the pronoun *he*. In this sentence, the correct correspondence is between *he* and *John* not between *he* and *Mary*.

4. References

1. Bright, D. and Vernik, R. (2004) LiveSpaces: An Interactive Ubiquitous Workspace Architecture for the Enterprise *Embedded and Ubiquitous Computing, International Conference EUC 2004*. Vol. 3207, Aizu-Wakamatsu City, Japan, Yang, L. T., Guo, M., Gao, G. R.Jha, N. K. (eds.), Springer

2. Jurafsky, D. and Martin, J. H. (2000) *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall Series In Artificial Intelligence, Russell, S. and Norvig, P. ed. New Jersey, Prentice Hall

3. Jurafsky, D. and Martin, J. H. (2009) *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2nd ed. Prentice Hall Series In Artificial Intelligence, Russell, S. and Norvig, P. ed. New Jersey, Prentice Hall

4. Phillips, M. (2009) *Livespaces Technical Overview*. DSTO-TR-2188, Edinburgh, DSTO

5. Broughton, M., et al. (2002) Conversing with Franco, FOCAL's Virtual Adviser *Virtual Conversational Characters (VCC) Workshop, Human Factors Conference (HF2002)*, Melbourne, Australia

6. Estival, D., et al. (2003) Spoken Dialogue for Virtual Advisers in a semi-immersive Command and Control environment *4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan

7. Wark, S., et al. (2004) FOCAL: A Collaborative Multimodal Multimedia Display Environment *SimTecT*, Canberra

8. Rayner, M., Hockey, B. A. and Bouillon, P. (2006) *Putting Linguistics into Speech Recognition: The Regulus Grammar Compiler*. CSLI Studies in Computational Linguistics, Copestake, A. ed. Stanford, California, CSLI Publications

9. Cohen, P. and Oviatt, S. (1994) The Role of Voice in Human-Machine Communication. In: Roe, D. and Wilpon, J. (eds.) *Voice communication between humans and machines*. Washington D.C., National Academy Press 34-75

10. Margetts, A. and Austin, P. K. (2007) Three-participant events in the languages of the world: towards a cross-linguistic typology. *Linguistics* **45**

11. Littlefield, J. and Broughton, M. (2005) Dual-type automatic speech recogniser designs for spoken dialogue systems *Australasian Language Technology Workshop*, Sydney, Australia, Baldwin, T., Curran, J.Zaanen, M. v. (eds.)

12. van Zaanen, M. and Somers, H. (2005) DEMOCRAT: Deciding between Multiple Outputs Created by Automatic Translation *The Tenth Machine Translation Summit, Proceedings of Conference; Phuket*, Phuket, Thailand

13. van Zaanen, M., Broughton, M. and Littlefield, J. (to be published) Improving Automatic Speech Recognition Output by Majority Voting: to be published, Defence Science & Technology Organisation

Page classification: UNCLASSIFIED

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

A Spoken Dialogue System for Command and Control

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document	(U)
Title	(U)
Abstract	(U)

4. AUTHOR(S)

Adam Saulwick, Jason Littlefield and Michael Broughton

5. CORPORATE AUTHOR

DSTO
Defence Science and Technology Organisation
PO Box 1500
Edinburgh South Australia 5111 Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-TR-2754	AR-015-418	Technical Report	October 2012

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
2008/1010343/1	04-195	COMD 1 DIV	28	13

13. URL on the World Wide Web

http://dspace.dsto.defence.gov.au/dspace/

14. RELEASE AUTHORITY

Chief,
Command, Control, Communications and Intelligence Divisio

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS Yes

18. DSTO RESEARCH LIBRARY THESAURUS http://web-vic.dsto.defence.gov.au/workareas/library/resources/dsto_thesaurus.htm

Computational Linguistics, Natural Language Processing, Speech recognition, Spoken Dialogue Systems

19. ABSTRACT

A speaker-independent Spoken Dialogue Systems (SDS) is proposed as a more natural interface for human–computer interaction than the traditional point-and-click method. This report describes the objectives, development and initial implementation of an SDS within a prototype command environment. It includes design decisions and solutions to problems encountered during development as well as comments regarding ongoing and planned future research that have emerged through these activities.